P.B.5818 - Patentlaan 2
2280 HV Rijswijk (ZH)
☎ +31 70 340 2040
TX 31651 epo nl
FAX +31 70 340 3016

Europäisches Patentamt | European Patent Office | Office européen des brevets

Generaldirektion 1 | Directorate General 1 | Direction Générale 1

de Pena, Alain
Compagnie IBM France
Département de Propriété Intellectuelle
06610 La Gaude
FR

Datum/Date
04/05/01

Zeichen/Ref./Réf.	Anmeldung Nr./Application No./Demande n°./Patent Nr ./Patent No./Brevet n°.
FR920000078	01480007.2 2216

Anmelder/Applicant/Demandeur/Patentinhaber/Proprietor/Titulaire
INTERNATIONAL BUSINESS MACHINES CORPORATION

Übersendung von/Transmission of/Envoi de

Antrag vom/Request dated/Requête du 24/04/01

☐ Kopien bei Akteneinsicht nach Regel 94(3) EPÜ
Copies in the case of inspection of files pursuant to Rule 94(3) EPC
Copies en cas d'inspection publique selon la règle 94(3) CBE

☐ Beglaubigung
Certification
Certification

☒ 1 Prioritätsbeleg(e)/priority document(s)/document(s) de priorité R. 94(4)

☐ ____ Ausfertigung(en) der Patenturkunde nach Regel 54(2) EPÜ
Duplicate of the patent certificate pursuant to Rule 54(2) EPC
Duplicata du certificat de brevet, selon la Règle 54(2) CBE

☐ Auszug aus dem Register nach Regel 92(3) EPÜ
Extract from the register pursuant to Rule 92(3) EPC
Extrait du registre selon la Règle 92(3) CBE

☐ Auskunft aus den Akten nach Regel 95 EPÜ
Communication of information contained in the files pursuant to Rule 95 EPC
Communication d'informations contenues dans la dossier selon la Règle 95 CBE

☐ Akteneinsicht nach Regel 94(2) EPÜ
Inspection of files pursuant to Rule 94(2) EPC
Inspection publique selon la Règle 94(2) CBE

Europäisches Patentamt	European Patent Office	Office européen des brevets

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

01480007.2

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

I.L.C. HATTEN-HECKMAN

DEN HAAG, DEN
THE HAGUE, 04/05/01
LA HAYE, LE

This Page Blank (uspto)

Europäisches Patentamt
European Patent Office
Office européen des brevets

Blatt 2 der Bescheinigung
Sheet 2 of the certificate
Page 2 de l'attestation

Anmeldung Nr.:
Application no.: 01480007.2
Demande n°:

Anmeldetag:
Date of filing: 23/01/01
Date de dépôt:

Anmelder:
Applicant(s):
Demandeur(s):
INTERNATIONAL BUSINESS MACHINES CORPORATION
Armonk, NY 10504
UNITED STATES OF AMERICA

Bezeichnung der Erfindung:
Title of the invention:
Titre de l'invention:
A method and system for distribution of file update in a client-server enviroment

In Anspruch genommene Priorität(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)

Staat:	Tag:	Aktenzeichen:
State:	Date:	File no.
Pays:	Date:	Numéro de dépôt:

Internationale Patentklassifikation:
International Patent classification:
Classification internationale des brevets:
/

Am Anmeldetag benannte Vertragstaaten:
Contracting states designated at date of filing: AT/BE/CH/CY/DE/DK/ES/FI/FR/GB/GR/IE/IT/LI/LU/MC/NL/PT/SE/TR
Etats contractants désignés lors du depôt:

Bemerkungen:
Remarks:
Remarques:

A METHOD AND SYSTEM FOR DISTRIBUTION OF FILE UPDATE IN A CLIENT-SERVER ENVIRONMENT

Field of the Invention

The present invention generally relates to data distribution in a client server environment and more particularly this invention applies to transfer of data update through a computer network.

Background of the Invention

In a client-server environment, IT resources are managed by a comprehensive solution including features such as network management as well as application management.

Application management on distributed sites implies code installation and update. To keep applications available, new versions of software need to be distributed through the network and installed on the target computers.

For instance, with the use of Tivoli (Tivoli is a trademark of Tivoli Systems Inc. in certain countries) a software distribution system, customers can rapidly and efficiently deploy mission-critical or desktop productivity applications to multiple locations from a central point. With such software distribution systems, the administrator builds a software package to be distributed from the management server to the clients, more precisely, from the management server to the code subscribers on the end point stations. A software distribution system uses a protocol for software distribution. This protocol is implemented both in the management server, some specific nodes of the network and the end point stations.

The software packages files are built on the software manager server. They contain the new code to be installed

directives for installation understandable by the receiving end points.

The software package files are then sent from the administrator console connected to the software manager server through the network to the subscribers. A software distribution system may implement in intermediate nodes applications for efficiently routing software packages according to the list of subscribers. These intermediate nodes are called gateway with Tivoli.

The end point stations are able to receive the software package files sent through the network and to install the corresponding new version of the software. An application, so called software distribution agent, operates on the end point stations for installation of the software and for applying the appropriate configuration changes to the system configuration.

The load of the network, in a distributed environment must always be minimized. Even if the technique of gateways for routing software packages already improves the use of bandwidth on the network lines, the size of the software packages remains critical. There is a need for minimizing the use of bandwidth for the download of software package files sent from the management server to the end point stations.

Prior art solutions in this area concentrate on changes at the file level. With WebDAV from Microsoft corporation, the current version to be sent is checked against the previous version and, if a file has changed, it needs to be transmitted, if not, then, transmission is not required. It is also common to group the changed files with the installation commands and to compress them before sending the package over the network.

In the US patent 5,721,907, the approach for solving the same problem is, rather than transferring the entire source file, to identify the differences between the previous files to be updated and the new files. Only the differences are transferred to the end point stations. The solution disclosed in the US patent consists in dividing the source into same size blocks and assigning to each block a computed key reflecting if there was a change or not in the corresponding block of data. The key computing is performed in both the receiving and the sending computer. A communication dialog is established between the sending and receiving computers, the result being that only blocks having a different computed key are sent from one computer to the other.

The principle of sending only the updates should be improved to fit with an existing framework for software distribution applying to a client-server environment. The solution of prior art rather applies to a communication between two computers connected through a communication line. As a matter of fact, it is not possible in a client-server environment to establish a protocol dialog between the sending and receiving computers as the sending is done between one software manager server and a thousand of end point stations.

There is a need for a solution to sending in once and in a secure way, software packages including only code updates.

Summary of the Invention

The invention consists in a method for distributing a data file, which is a modified data file of a base data file, as a distribution package file in a data file distribution system comprising a distribution server, where the distribution package file is created, a network, comprising nodes adapted for routing the distribution package file to end point stations which are themselves adapted to install the

distribution package file, said method comprising the steps of creating, on the distribution server, a distribution package file, the delta distribution package file, comprising the delta file, result of a differencing algorithm applied to the base data file and the modified base data file and a data integrity code applied to the base file, and, receiving, on the end point stations storing the base data file, said delta distribution package file, comparing the data integrity code with the data integrity code of the stored base data file and, if the code is identical, reading the delta file and rebuilding the modified data file from the base data file and the delta file.

The method of the present invention may also have the step of creating the delta package comprising a step of writing in the delta file at least one byte block itself comprising one directive for copying a sequence of bytes from the stored base data file and the byte offsets identifying said sequence in said base data file and wherein the rebuilding step further comprises a step of copying using the byte offsets, when one directive for copying is read in the delta file, said sequence of bytes from stored base file to the rebuilt modified data file.

The method of the invention may also include in the step of creating the delta package, a step of writing in the delta file at least one byte block comprising one directive for adding a new sequence of bytes and the new sequence of bytes, and the rebuilding step further comprises a step of copying, when one directive for adding is read in the delta file, said new sequence of bytes to the rebuilt modified data file.

One major advantage of the current solution is that it applies to a byte level, this means that the method is applicable not only to the distribution of applications but also to the distribution of data files. For instance, this

method can apply to the distribution of price lists which need to be periodically updated on thousand of workstations. With the method of the present invention, one can generate a "delta file" that contains only "changed prices" between the previous list and the current one. This method is not dependent on the format of the data files to be compared and updated.

Brief Description of the Drawings

FIG. 1 is an illustration of the software distribution system wherein the solution of the present invention may be implemented;

FIG. 2 illustrates the content of the code update file obtained by the method according to the present invention;

FIG. 3 is an example of the optional "depot table" which keeps track of the software packages which are stored in a depot close to the endpoint station.

Fig. 4 shows the flow chart of the method for building the software package on the software manager server according to the present invention;

Fig. 5 shows the flow chart of the method for receiving and installing the software package on the end point stations according to the present invention.

Description of the preferred embodiment

Figure 1 shows the software distribution process in a client server environment such as the Tivoli software distribution system. The administrator (100) accesses the software manager server (110) to prepare the software packages, send them through the network and ask for their installation in the system libraries of the end point stations corresponding to a list of subscribers. A software package is

a file containing the new code to be installed and directives for new code installation to be executed on the end point stations. The new code may comprise one or more than one file. The administrator creates software package files on the software manager server. The administrator uses the user interface, preferably a graphic user interface, with the software distribution application operating on the software manager server. The distribution of software package is activated by a command from the software manager server (110). The server sends the software package files to the designated target end point stations through the network (120). According to Tivoli software distribution system, gateways are used as intermediate routing points for the software distribution. In Figure 1, the gateway (130) is able to identify that the software distribution package file to be sent to a list of subscribers must be distributed to three end point stations (140, 150). The software package file is routed by the gateway to the target end points which are either personal computers (140) or other servers (150). Once received in the target end points, the software package is read and launched by an application, it may be the management agent of Tivoli, for code installation and execution of system library update.

As described in Figure 1, the software is distributed by one command from the server to the designated end point stations. The launching of the code update may be started at a differed time according to the sophistication of the management agent application. With the use of the preferred embodiment as implemented in the server (110) and in the end points (140, 150) the distribution is done in once but the size of the code is dramatically reduced as it only conveys a delta software package file only comprising code updates.

Figure 2 illustrates the resulting code update file comprising the encoded software update according to the method of the preferred embodiment. The base file (200) is the

previous version of the code to be updated. The version file (210) is the new version of the code which needs to be installed and run on the end point stations. The delta file (220) is the real file which will be sent, result of the method of the preferred embodiment. The delta file is a succession of blocks which can be of two types: blocks comprising "matching sequence" of code and blocks comprising new sequences of code. Between the base file and the version file some matching sequences of bytes are identified (205, 206). These sequences are common to the base file and the version file. These are the part of the code which is common to the previous and the new version of the code. The matching sequences of code will not be copied in the delta file with the method of the preferred embodiment. Only the new data (207, 208 and 209) will be part of the resulting delta file. Each block of the delta file comprises directives (225, 230) directly executable by the end point stations. The two directives used (225, 230) in the delta file are the "add" command (225) preceding the code portions which are new and the "copy" command (230) preceding the code portions to be copied from the previous version of the code to be updated (205, 206). The "copy" command has parameters providing the offset of the field to be copied in the version file. The delta file can be read on the end point stations which will be able to rebuild the new version to be installed.

The software distribution process of Tivoli is accomplished in three phases: the first phase is for preparing the software package corresponding to a set of new code to be distributed and installed on end point stations. The delta file for each new code file is prepared. The software package file comprises the delta files and in its header, crc32 of the base file for each delta file. This data integrity code computed at the creation of the software package is used by the end point station to check the validity of the base files before starting installation of the new code with the delta

files. This software package is built by the administrator from a console connected to the software manager server. The administrator starts an application operating on the software manager server using the graphic user interface for entering commands. The first phase is started with the "Build SP" command. This command starts the building of the software package file. Parameters such as the name and version of the software package (SP_LABEL, SP_VER) are provided to the application by the administrator during the first phase. The optional depot process in Tivoli software distribution system is used when software are frequently updated. The software package is installed on a depot close to the end point station. A depot is a gateway configured in such a way to cache software packages so that they don't need to be re-transmitted from the server any time they are distributed, saving network bandwidth.

The installation is performed on the end point station using the software distribution package installed on the close station: this process helps in offloading the end point station from storing the software packages. If the optional depot process is used, the name of the software distribution (DEPOT_LABEL for a depot) is seized as a parameter of the Build SP command. With the optional depot process a tracking of software distribution operations is kept on the software manager server.

The second phase of the software distribution consists in sending the software package to a list of end point stations, which can be either personal computers or servers, and on which the new version of the code needs to be installed. The sending command of software packages, which is the downloading with Tivoli software distribution, is performed through the administrator application. The sending is executed either immediately or it can be delayed. With Tivoli software distribution, the software package can be sent through the

network either directly to a end point station or to the software distribution gateways as described in Figure 1 which themselves route the software distribution package to the end point stations. When the download is executed the download timestamp is stored by the application on the software manager server. If the option of software depot is chosen, the software packages are sent to a gateway close to the end point station for a further process of new code installation from this close station.

The third phase of the software distribution process is executed on the end point stations which have, with an option in Tivoli, a software distribution agent able to receive the software package, read it and launch the installation process. These operations are performed sequentially on the end point stations. Optionally, these operations can be separately executed. Even if it is possible to delayed all the intermediate operations, the process of sending software and installing it on the end point stations is usually started from the administrator console when the command "SD INSTALL" is entered.

The software distribution method of the preferred embodiment is implemented in these three phases of the software distribution process. With the method of the preferred embodiment in the first phase, the administrator application allows either a "Build SP" or a "Build delta SP". Two additional parameters are given to the application by the administrator in the preferred embodiment. They are the type of software distribution (TYPE) and the name and version of the previous code to be updated (BASE_SP_NAME, BASE_SP_VER).

Figure 3 shows an example of the new "depot table" built during the first phase of the method of the preferred embodiment when the software depot option is chosen by the administrator. This table, part of the depot option in Tivoli,

is used to keep track of the different creation and downloading of code when they are frequent. The table, stored on the software manager server, is filled by the application building the software package on the software manager server. The table stores the parameters entered by the administrator which are used for the software distribution and the time of downloading of the software package. In Figure 3, the first row describes the depot_1 "depot" which is the software package for installing the application myapp version 2.0 as a delta software package using as a base file the application myapp version 1.0, timestamp being the time of the downloading of the delta software package. The second row records a second creation of software package which corresponds to the depot depot_2, of myapp version 2.0. This second row is for a full software package for this application and version downloaded at the timestamp as stored.

Figure 4 shows the flow chart of the steps of the method of the preferred embodiment for building the software package. In the preferred embodiment an application is executing on the software manager server to which the administrator console is connected. Through a graphic user interface the administrator can order the building of a software package and provides parameters. If the optional depot option is chosen, the operations on the software packages are all recorded. This can be very useful for application codes frequently updated and distributed to the end point stations. The end point station receiving the application code updates are the "subscribers". The parameters entered by the administrator are the name and version of the application code to be sent, the list of names of the subscribers. The software package comprises both code and commands to be executed on the and point station which allow launching the code installation. These commands are machine type dependent and thus the final software package file which will be downloaded will be machine type dependent. The type of software package to be built will be defined by

the application according to the subscriber name entered by the administrator. Once the parameters are entered (400), the application asks the administrator if he wants to build a delta software package (405). If the answer to the test (405) is no, the application builds a software package (410) comprising all the new code that is without reducing the size of the software package file. The software package comprises also a directive to install all the following code which is the "add" command understandable by the end point station. If the answer to the test (405) is yes, a delta software package file is to be built. A differencing algorithm is then performed on the file containing the previous version of the code and the new code. The differencing algorithms, known from the prior art, find and provide in output the differences between a file and a modified version of the same file. In output, the differencing algorithm provides a delta file as described above in Figure 2. The input of a differencing algorithm is the two versions of the code file which are the base file and the version file. The delta file is a sequence of directives add and copy. The add directive contains new data that must be added to the base file at a certain offset to rebuilt the version file; the copy directive only indicates what data bytes are to be copied to the version file to rebuild it. The delta file is a compressed version of the version file with the constraint that it needs the base file to rebuild the version file. Coming back to Figure 4, using a differencing algorithm, the identification of differences in the code is started (415). If the end of file is not reached (answer no to test 420), and if one matching sequence is identified between the base code and the new version (answer yes to test 425), a "add" directive is written (435) on the delta file, output of the differencing algorithm. The add directive identifies the offsets where code is to be copied from the base file to rebuild the new version of the code from the delta file and the base code. If no matching sequence is identified between the base code and the new version (answer

no to test 425), the part of new code coming from the new version is copied to the delta file with the "add and copy" directive (430) for adding the new code copied for rebuilding the new version from the delta file. If the end of file is reached (answer yes to test 420) the CRC32 of the base file is added (440) to the header of the software package file. This CRC32 or any data integrity checking code is used to insure, at the end point station, the use of the correct base file to start rebuilding the new version starting from the base file supposed being already installed in the end point station. As the CRC32 is located in the header of the software package, the checking of data integrity is performed before reading the code update. If the base file located on the end point station has the same CRC32 than the code written in the header of the software package the installation process goes on. If not, the operation is abandoned. There are as many CRC32 in the header of the software package than the number of base files to be used in the installation of the new code.

If the depot option is used, the depot table keeping track of the created and downloaded code updates is updated (445) with the inputs of the administrator as described in the first row of the table of Fig. 3. This step of the method is not mandatory. If the choice of the administrator is to download the entire new version of the code without using the advantage of the delta file, (answer no to test 405) the software package file is built with directives for adding the entire new code which is copied after the add and copy directive (410). If the depot option is used, the depot table is then updated as described in the second row of the table of Fig. 3.

The method comprises two steps for downloading a software package file built according to the previous steps of the method. The first step consists in sending the software package file to the "subscribers" as listed in the parameters

provided by the administrator. The software package will be adapted to the system operating on the end point station of the subscriber. The telecommunication protocol is network dependent. The software package to be sent can be sent to a gateway acting as intermediate software distribution node according to the software distribution architecture which is Tivoli in the preferred embodiment. The second step after sending is, if the depot option is used, the update of the depot table with the downloading timestamp written in the last field of the table row as described with Fig. 3.

The flow chart of Figure 5 shows the steps of the method of the preferred embodiment for installing a new version of a code on an endpoint station. If the depot option is used, the software package is stored intermediately on a gateway close to the end point station. If the depot option is not used, the software package is sent directly from the software manager server to the end point station through the network. The new version of the code will be installed using a software package prepared by the steps of the method as described in Figure 3 and received by the end point station. A software distribution package installation may be started from the administrator console or from the end point station itself. With the corresponding command "SD_INSTALL", is provided, as attribute, the name of the software package which can be either a "full" software package containing the entire new code to be installed or a "delta" software package containing a delta file. The installation of a "delta" software package consists in rebuilding the new version of the code from the base file (the previous version of the code) already installed on the end point station. When a "SD_INSTALL" command is started (500), if a delta software package is to be installed (answer yes to test 510), the CRC's the header of the software package is extracted and checked against the CRC's the base files of the previous level of the code which are already stored on the end point station. If the checking is not correct (answer no

to test 520), the SD_INSTALL operation is stopped with an error message (525) saying that the base file for the corresponding new code files is or are not correct and that the new version of the code for this file or these files cannot be rebuilt from the base file or base files stored on the end point station. If the CRC checking is correct (answer yes to test 520), the reconstruction process is launched and an output file containing the rebuilt new version of the code is prepared for each delta file received. The delta files are sequentially read. When a add directive is encountered, there is a matching sequence (answer yes to test 540), of bytes identified in the delta file by the offsets in the base file. This matching sequence is extracted from the base file and copied (550) to the output file. When an add copy directive is encountered, there is a sequence of bytes which is new vis a vis the previous version of the code and the following bytes in the delta file are copied (545) to the output file. This sequence of steps is repeated while the end of delta file is not reached (answer No to test 520) and for each of the delta files contained in the software package.

If the SD_INSTALL command specifies the installation of a "full" software package (answer no to test 510), the entire code stored in the software package files is copied on the end point station (535).

Once the new version is installed after copying the entire code stored in the "full" software package or when the end of delta file has been reached (answer yes to test 530) for a "delta" software package, the system libraries are updated with the references to the new version of the code (560) and the operation ends (565).

More commonly, the installation operation is implemented as a program operating on the end point station. It is activated as a command from one other program operating on the

same end point station or from one other program operating on the software manager server. This later program is part of the application operating on the software manager server accessed via a graphic user interface, in the preferred embodiment, from the administrator console.

It is noted that the method of the preferred embodiment requires computing resources on the software manager server to build the delta software package and on the end point stations to rebuild the new version of the code from the delta software package. More particularly, the appropriate differencing algorithm should be used to minimize memory requirement and CPU time as recommended in the thesis "Differential completion: A Generalized Solution for Binary Files" in completion of the Master's of Science degree, Department of Computer Science, University of California, Santa Cruz, December 1997. For small files one can use the HPCP algorithm while for bigger files (greater than 10 Mb), it is more appropriate to use One Pass algorithm. HPCP and the One Pass algorithm are both described in the document referenced sooner.

The same method as described may be applied to the distribution of an updated version of any existing byte data file because it applies at the byte level. The method applied to data file distribution provides the same advantage of line bandwidth saving in the network used for distribution.

CRC32 or any type of CRC or any code used for data integrity checking known by the man skilled in the art can be used in the method of the present invention for data integrity checking of the base file and for the security of the rebuilding operation of the new data file in the end point stations.

Claims

1. A method for distributing a data file, which is a modified byte based data file of a base byte based data file, as a distribution package file in a data file distribution system comprising a distribution server, where the distribution package file is created, a network, comprising nodes adapted for routing the distribution package file to end point stations which are themselves adapted to install the distribution package file, said method comprising the steps of:

creating, on the distribution server, a distribution package file, the delta distribution package file, comprising a delta file, result of a differencing algorithm applied to the base data file and the modified data file, and a data integrity code applied to the base file;
and,

receiving, on the end point stations storing the base data file, said delta distribution package file, comparing the data integrity code with the data integrity code of the stored base data file and, if the code is identical, reading the delta file and rebuilding the modified data file from the base data file and the delta file.

2. The method of claim 1 wherein the step of creating the delta package comprises a step of writing in the delta file at least one byte block comprising one directive for copying a sequence of bytes from the stored base data file and the byte offsets identifying said sequence in said base data file and wherein,

the rebuilding step further comprises a step of copying, using the byte offsets, when one directive for copying is read in

the delta file, said sequence of bytes from stored base file to the rebuilt modified data file.

3. The method of claim 1 or 2 wherein the step of creating the delta package comprises a step of writing in the delta file at least one byte block comprising one directive for adding a new sequence of bytes and the new sequence of bytes, and wherein,

the rebuilding step further comprises a step of copying, when one directive for adding is read in the delta file, said new sequence of bytes to the rebuilt modified data file.

4. The method of anyone of claims 1 to 3 wherein the differencing algorithm is the HPCP algorithm.

5. The method of anyone of claims 1 to 4 wherein the data integrity code is a CRC32.

6. A system for distributing a data file, which is a modified data file of a base data file, as a distribution package file, said system comprising a distribution server, where the distribution package file is created, a network, comprising nodes adapted for routing the distribution package file to at least one end point station which are themselves adapted to install the distribution package file, said system being characterized in that:

the distribution server comprises means for creating a delta distribution package file, comprising the delta file, result of a differencing algorithm applied to the base data file and the modified data file, and a data integrity code computed from the base file;
and,

the end point station stores the base data file and comprises means for receiving said delta distribution package file, comparing the data integrity code with the data integrity code of the stored base data file and, if the code is identical, reading the delta file and rebuilding the modified data file from the base data file and the delta file.

7. The system of claim 6 wherein the distribution server further comprises means for writing in the delta file at least one byte block comprising one directive for copying a sequence of bytes from the stored base data file and the byte offsets identifying said sequence in said base data file and wherein,

the end point station further comprises means for copying, using the byte offsets, when one directive for copying is read in the delta file, said sequence of bytes from stored base file to the rebuilt modified data file.

8. The system of claims 6 or 7 wherein distribution server further comprises means for writing in the delta file at least one byte block comprising one directive for adding a new sequence of bytes and the new sequence of bytes, and wherein,

the end point station further comprises means for copying, when one directive for adding is read in the delta file, said new sequence of bytes to the rebuilt modified data file.

9. The system of anyone of claims 6 to 8 wherein the differencing algorithm is the HPCP algorithm.

10. The system of anyone of claims 6 to 9 wherein the data integrity code is a CRC32.

Abstract

A METHOD AND SYSTEM FOR DISTRIBUTION OF FILE UPDATE IN A CLIENT-SERVER ENVIRONMENT

A method and system for distributing byte data files to end point stations through a network, the byte data files being modified versions of an base byte data file stored on the end point stations. The method comprises steps for creating a delta software package comprising at least one resulting delta file obtained by applying a differencing algorithm to the base byte data file and the modified byte data file. The method further comprises the step of adding in the header of the software package file, a data integrity code of the base byte data file. On the end point stations, the method comprises steps for comparing the base byte data file integrity of the code in the delta package to the code in the base byte data file stored on the end stations. If the code is identical, the delta file is used to rebuild the modified version of the base byte data file from the base byte data file stored on the end stations.

Figure 3.

This Page Blank (uspto)

FR9 2000 0078
Crudele et al.
1 / 5

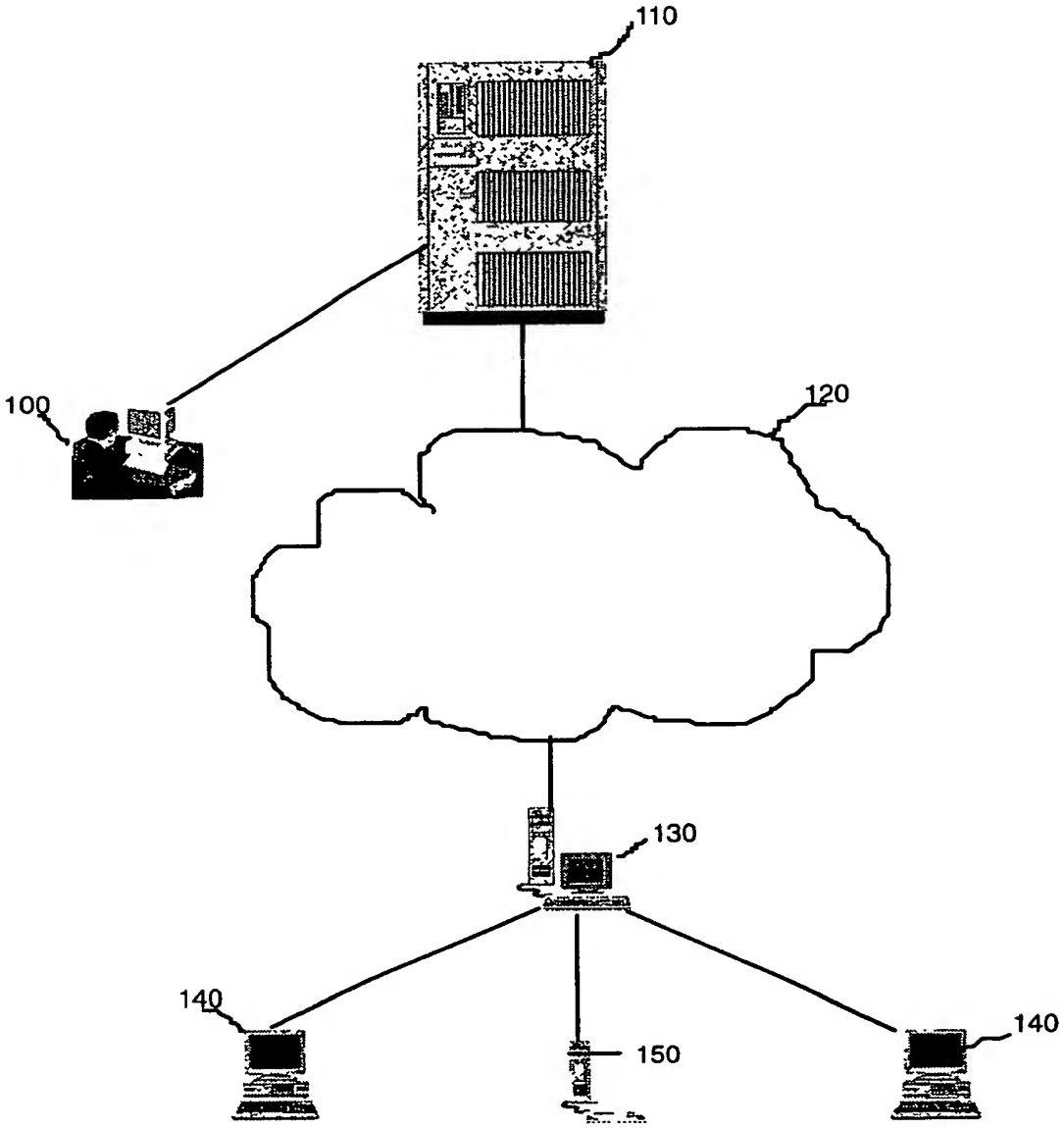

FIGURE 1

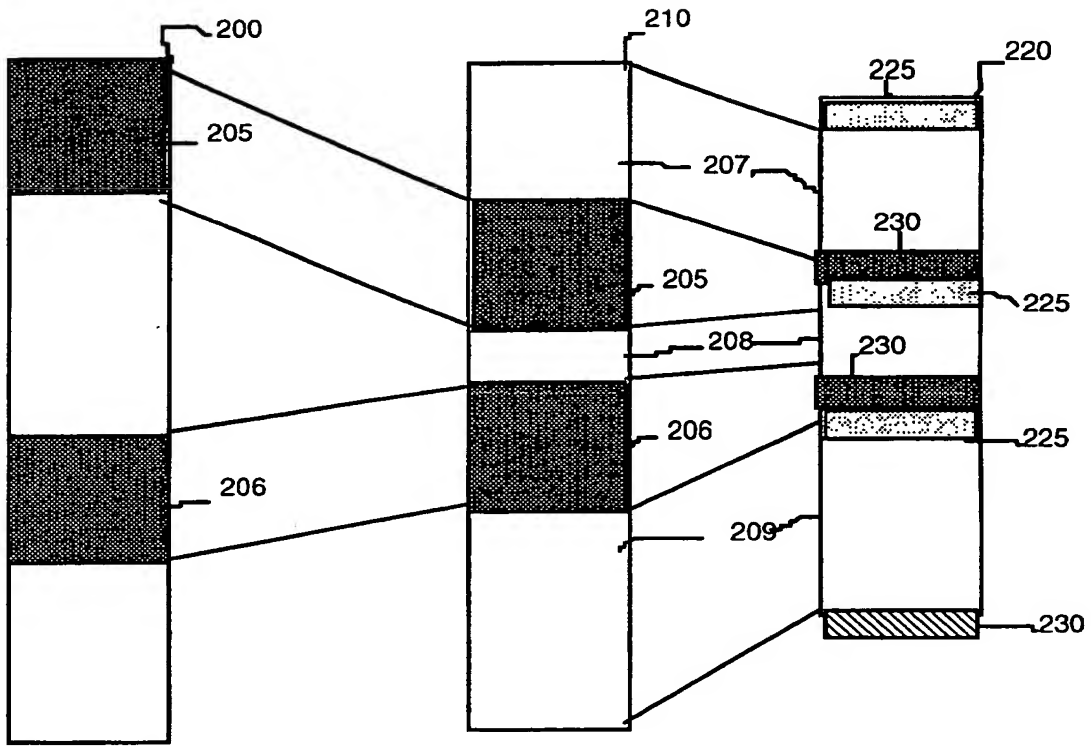

FIGURE 2

DEPOT	SP_NAME	SP_VER	TYPE	BASE_SP_NAME	BASE_SP_VER	LOAD TIME
depot_1	myapp	2.0	DELTA	myapp	1.0	timestamp
depot_2	myapp	2.0	FULL			timestamp

FIGURE 3

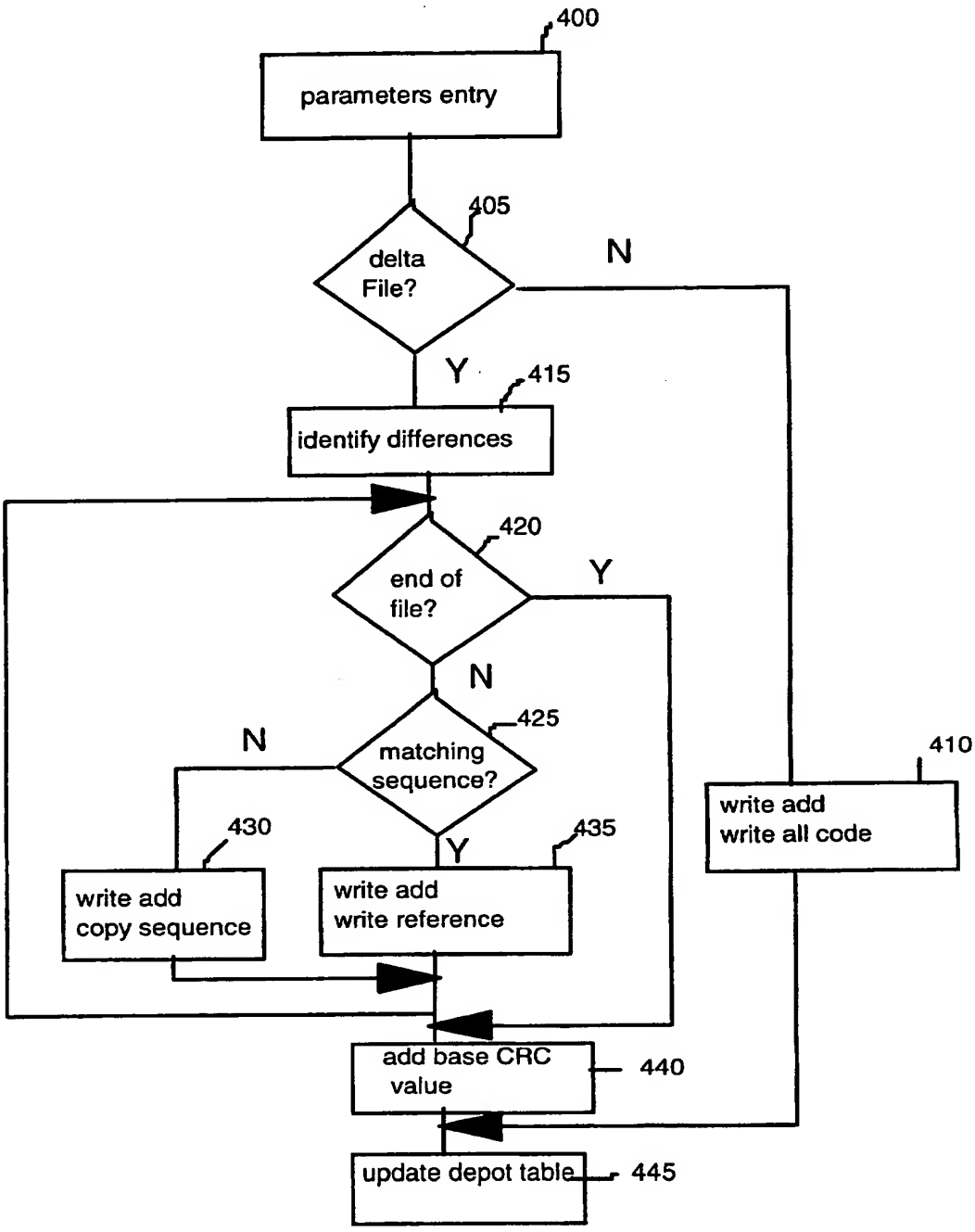
400
parameters entry
405
delta File?
N
Y
415
identify differences
420
end of file?
Y
N
425
matching sequence?
N
Y
410
write add write all code
430
write add copy sequence
435
write add write reference
add base CRC value
440
update depot table
445

FIGURE 4

FR9 2000 0078
Crudele et al.
5 / 5

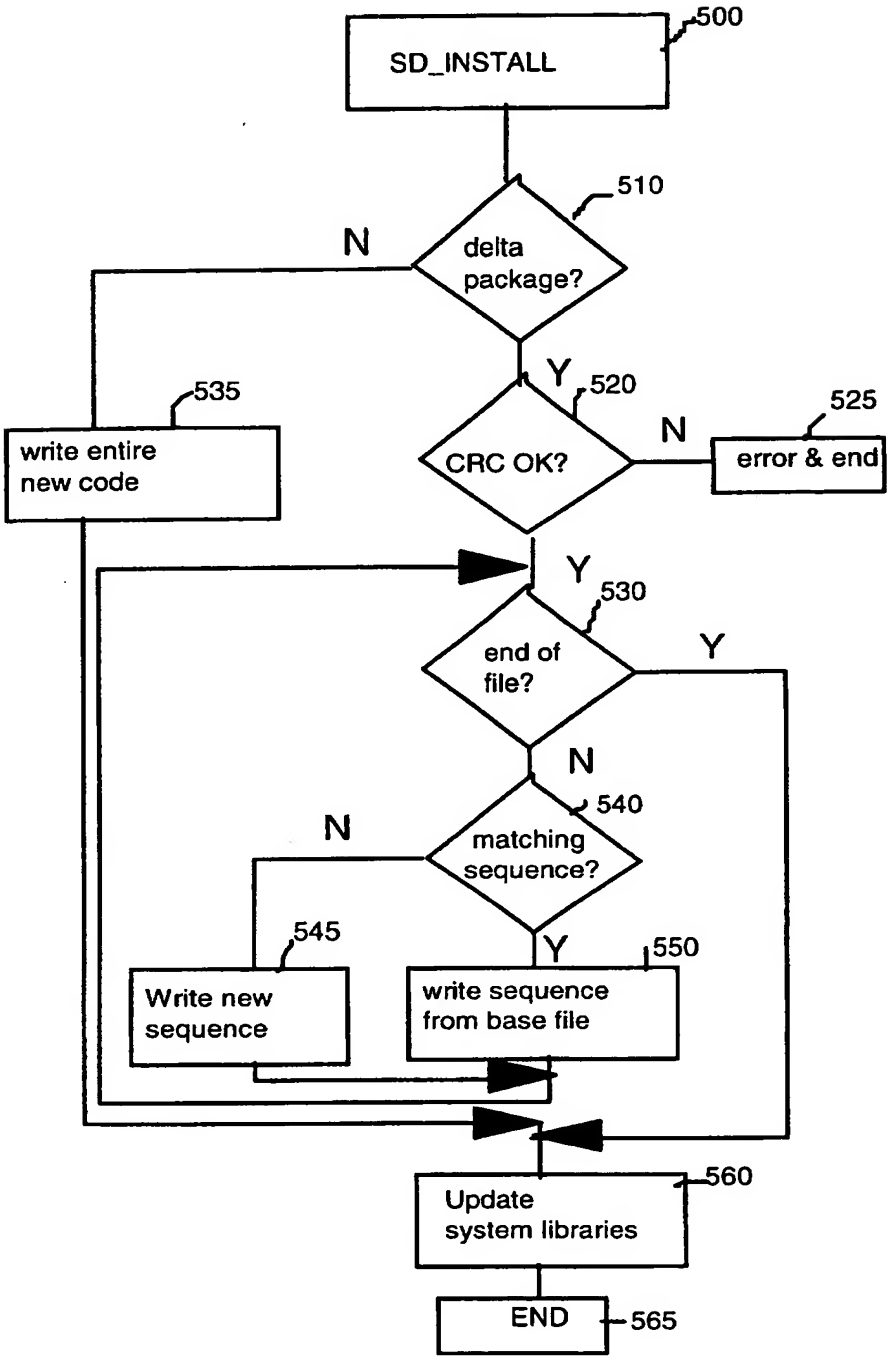

FIGURE 5